F.I.Z.O.F.

FLOPPY INTERFACE

FOR

ZX81

OPERATING FIRMWARE

USER MANUAL

CENTRONIC INC.
1101 Bristol Road
Mountainside, New Jersey 07092
Telephone: (201) 233-7200 • Telex 138275

CONTENTS

1. OVERVIEW
2. FIZOF System Variables
3. FIZOF Command Routines
4. Initiallising the FIZ System
5. Preparing a New Diskette
6. Saving a BASIC Programme
7. Loading a BASIC Programme
8. Producing a Directory Listing
9. Making a Status Enquiry
10. Creating a New Data File
11. Writing a Random Access Record
12. Reading a Random Access Record
13. Removing a File From The Disk
14. Appendix A

FLOPPY INTERFACE FOR ZX81 - OPERATING FIRMWARE V1.0

OVERVIEW

Disk storage is organised as 34 (39) data tracks of 10 (hard) sectors per track, each of 128 bytes, plus one directory track:-

```
Track Ø           - Directory Track
Track 1           - First Data Track (10x128 bytes)
  "                   "    "     "     "  "    "
  "                   "    "     "     "  "    "
Track 34 (39)     - Last Data Track  ( "  "    "  )
```

(figures for 40 track drive systems are given in brackets)

The Directory Track holds the Disk Name and a directory entry for each data track which contains a Track Code to indicate a full, empty or bad track and a File name / Type.

A File consists of a number of whole tracks so that file lengths may range from 1 x 10 x 128 bytes to 34 (39) x 10 x 128 bytes in steps of 1280 bytes. It follows that up to 34 (39) different programs or data files could be stored on one diskette if none were more than 1.25k long.

128
390
11520
384
49920

A number of Reserved Variables are needed for the FIZ systems to operate, these are:-

```
FIZOF       ( F$       Current File
System      ( R        Current Record No./No of records
                       for CREATE
Variables   ( R$(128)  Data Record

Addresses of     (INIT    DSAVE  DLOAD  DIR  STAT
Command Routines (CREATE  WRITE  READ   KILL NEWD
```

- these Variable Names should not appear in a program in any other context.

The FIZOF initiallisation routine automatically loads all the above Reserved Variables into the ZX81 BASIC Variables area in RAM, causing a memory overhead of approximately 240 bytes. The Command Routines have mnemonic variable names for operator convenience - the memory overhead would be reduced if, after initiallisation, the variables were cleared and F$, R, and R$ (128) entered explicitly. The Command Routines could then be accessed by decimal addresses as required.

1024
48
8192
4096
49152

APPENDIX A

FIZOF V1.0 ERROR CODES.

E =	Meaning:
0	No Error
1	System already initiallised (F$ exists)
2	System not initiallised
3	Bad Disk (Cannot write directory)
4	Write Protected (Notch covered.)
5	Write Error (Read after Write Verify fails repeatedly)
6	Hard Read Error (Sector may be permanently unusable)
7	Bad Filenames (wrong format)
8	No room on disk
9	File already exists on this disk
10	File not found
11	No such record number.

Code is in
8*1024 to 10431

Reading a Random Access Record: READ, 10020

```
LET F$  =  "Filename.  Type"
LET R   =  Record No.
LET E   =  USR READ or PRINT USR READ
```

This is similar to WRITE except that in this case after the READ has executed successfully, R$ will contain the 128 bytes of data previously written to record number R.

Note that with FIZOF there is no opening or closing of files - all disk operations take place on the Current File whose name is held in F$.

String representations of numbers may need converting back to numeric variables, etc., e.g.

```
LET  A$  =  R$  (1  to 10)
LET  B$  =  R$  (11 to 20)
LET  C$  =  R$  (21 to 50)
.
.
Let  B   =  VAL (B$)
```

Removing a file from the disk: KILL, 10023

```
LET  F$  =  "Filename.  Type"
LET  E   =  USR KILL or PRINT USR KILL
```

This is the ONLY way of deleting a program or data file apart from using NEWD which deletes everything on the disk.
Use with caution - there is no second chance to stop the process.
Keep copies of important programs or data on separate disks or under "Filename. BAK" names.
(To copy data files - see appendix B.)

MAKE SURE that F$ contains the NAME OF THE FILE YOU ACTUALLY WISH TO KILL.

accord
catenatig
previous

Creating a New Data File: CREATE, 10014

LET F$ = "Filename. Type"
LET R = No. of sectors (records)
LET E = USR CREATE or PRINT USR CREATE

This command or statement creates directory entries for a new data file of given length. FIZOF will only handle fixed length files, to expand a file which proves to be too short it is necessary to CREATE a longer file under a different Name, copy the data across then KILL the first file.

The file length given by R is automatically rounded up to the next multiple of 10 sectors so that a file occupies a number of complete tracks. If the value held in R is less than 1 or greater than the number of free sectors on the disk, then Error Code E = 8 results.

Writing a Random Access Record: WRITE, 10017

LET F$ = "Filename. Type"
LET R = Record No.
LET R$ = "Record Contents"
LET E = USR WRITE or PRINT USR WRITE

F$ must hold the name of a previously CREATEd Data File. R must be a record (sector) number in the range 1 to (Length of File given by STAT) if it is outside this range Error Code E = 11 results.
R$ is used to hold the actual string to be written as disk record, up to 128 bytes.
The "Record Contents" could be specified by concatenatig a number of data fields which have previously been dimensioned e.g.

```
DIM  A$ (10), B$ (10), C$ (30),.......
.
.
LET  B$  = STR$ (B)
.
.
LET  R$  = A$ + B$ + C$ +.............
```

This also shows how a numeric variable (B) had been converted to a string (B$) before being added to the Record Contents.

In a similar way the 128 byte record could be composed of several data records to maximise on the capacity available, i.e. A$, B$ and C$, etc; could be data records in their own right - only needing segregation after the 128 byte record is read from disk.

Loading a BASIC Program: DLOAD, 10005

LET F$ = "Filename. Type"
LET E = USR DLOAD or PRINT USR DLOAD

This needs little explanation except to say that loading proceeds at about twice the speed of saving. An Error Code E = 10 will result if the Program Name cannot be found on the disk.

Beware of trying to load Data Files - the system will not know the difference and will crash.

Producing a Directory Listing: DIR, 10008

LET E = USR DIR

This command clears the screen and produces a display which shows the Disk Name at the top of the screen, followed by a list of Filename. Type for every Program or Data File on the disk. The order in which they appear is that of the last track associated with each file.

The DIR command has no affect on any program which happens to be in memory at the time.

Making a Status Enquiry: STAT, 10011

LET F$ = "" or LET F$ = "Filename. Type"
LET E = USR STAT

This command clears the screen and prints the Disk Name at the top. If Null String has been assigned to F$ then a Disk Status report results. This simply shows the number of free sectors on the disk. (each sector is 128 bytes)
If F$ holds the name of a file on the disk then that File Name will be printed followed by its length in sectors.

Beware of using PRINT USR STAT - it will add another zero to the number of sectors.

Smithsonian Institution
Libraries

Preparing a New Diskette: NEWD, 10026

LET R$ = "New Diskette Name"
LET E = USR NEWD or PRINT USR NEWD

This command or statement is used to erase the former contents of a diskette and prepare it for use with the FIZ system. A name to identify the new diskette must be assigned to R$ before NEWD is executed. This name may be up to 128 characters long and will appear at the top of DIRectory and STATus screens.

The NEWD process is to write the New Diskette Name to track 0, sector 0 then write a blank directory to the disk. A test pattern is then written to every sector of tracks 1 to 34 (39).

If the write/read/compare process fails on any sector then the whole track is declared Bad and removed from the Directory. To find out how much of the disk is usable after executing NEWD use the STAT command with a null string assigned to F$.

If the directory track (track 0) is unusable the Error Code E = 3 results and the diskette must be cleaned or discarded.

Saving a BASIC Program: DSAVE, 10002

LET F$ = "Filename. Type"
LET E = USR DSAVE or PRINT USR DSAVE

This command or statement saves a program on disk in a similar way to saving a program on cassette except the F$ must hold a unique Program Name in the proper format and, of course, the whole process proceeds much faster - about 2.5 seconds per 1k including the time taken to read back and verify each sector written.

"Load and go" operation is produced, as for cassette tape, by executing the DSAVE statement in a running program, e.g.

```
10  LET E  = USR 9999
20  LET F$ = "TEST. BAS"
30  LET E  = USR DSAVE
40  IF  E    0 THEN PRINT "DISK ERROR";E
50  PRINT "THIS PRINTS AFTER LOADING"
60  STOP
```

The programme Name will be added to the Disk Directory. If the name is not unique then an Error Code E = 9 will result and the program will not be saved:

AN EXISTING PROGRAM OF THE SAME NAME WILL NOT BE OVERWRITTEN, it has to be KILLed explicitly.

FIZOF Command Routines

All FIZOF commands may either be actioned from the keyboard or used in program statements.

They take the form:-

LET E = USR Command Mnemonic

When control returns to BASIC the variable E will contain an Error Code (listed as appendix (a).

A value of zero indicates that the command was executed successfully, and a program should always check for this condition.

With some commands entered from the keyboard it is convenient to use the form:-

PRINT USR Command Mnemonic

and this will print the error code on the screen for immediate inspection - but this may be confusing when used with DIR or STAT.

Initiallising the FIZ System: INIT, 9999

LET E = USR 9999 or PRINT USR 9999

After the FIZ system components are connected and the system is switched on, one of the above commands must be given before disk operations may be executed. Note that in this case the Numeric address is always used to initiallise the system, since until this is done the mnemonic form cannot be recognised, although subsequently it could be useful in ensuring that "rest state" conditions are selected.

LET E = USR 9999 should be the first statement line in any program running under the FIZ system otherwise when the program is run, all the Reserved - Variables would be cleared from memory. The effect of this statement is to select "rest state" conditions on the disk interface lines and to load the Reserved Variables into the BASIC system. A check is made to see if the variable F$ already exists, if it does FIZOF assumes that initiallisation has already been done and an Error Code E = 1 is returned or printed.

FIZOF System Variables

F$ CURRENT FILE, Variable length string, initiallised to Zero length.
-holds the Name and/or Type of the file that FIZOF will access when a command is executed. The format is:-

Filename	-	Ø to 6 Non-space characters, not "."
Separator	-	" " (Full Stop)
File Type	-	Ø to 3 Non-space characters, not "."

File Name and Type must not both be absent. Names and Types will be truncated to fit and any spaces ignored. File Types such as BAS, DAT etc may be used as an aid to the operator but have no significance to FIZOF.

R CURRENT RECORD/No. OF RECORDS, Numeric variable, (not the control variable of a FOR.....NEXT loop) initiallised to zero.
-used with Data Files, when creating a new Data File the operator or program assigns the required number of records to R. (this is then rounded up to the next multiple of 10).
When reading or writing an individual Data Record, R must hold the Current Record No. to define the record which will be accessed by the READ or WRITE commands.

R$ DATA RECORD, 128 byte String array, initial contents non-significant.
-used to hold the name of a new diskette before executing NEWD.
When reading from or writing to a Data File R$ contains 128 bytes of data, the contents of one physical sector.